De um módulo, só precisamos saber o seu *nome*, a sua *função* (**o que ele faz**) e as suas *interfaces*, isto é, o que ele precisa receber para funcionar e o que devolve depois de executada a sua função.

Um algoritmo modular tem as seguintes vantagens:

– Facilita o projeto, na medida em que permite a abstração.

– Permite a divisão de tarefas em uma equipe, onde cada módulo pode ser desenvolvido por um programador diferente.

– Facilita o teste, porque cada módulo pode ser testado individualmente, e depois ser incorporado ao conjunto.

– Maior flexibilidade e facilidade de manutenção, já que o módulo pode "esconder" dos outros módulos detalhes que só interessam a ele.

8.2 EXEMPLOS

PROBLEMA:

Colocar em ordem crescente os valores de uma tabela unidimensional de *n* entradas. O tamanho da tabela e os seus valores são obtidos de cartões. A tabela original e a tabela classificada devem ser emitidas.

SOLUÇÃO:

```
início {passo 1}
|   "ordenar"
fim;
```

```
refinamento de "ordenar";
|   "obter a tabela original e imprimir";
|   "ordenar a tabela";
|   "imprimir a tabela ordenada";
fim refinamento;
```

```
refinamento de "obter tabela original e imprimir";
|   leia (N);
|   para I de 1 até N faça
|   |   leia (TABELA [I]);
|   |   imprima (TABELA [I]);
|   fim para;
fim refinamento;
```



```
refinamento de "ordenar a tabela";
|   I ← 1;
|   enquanto I ≤ N − 1 faça
|   |   "procurar o menor elemento de TABELA [I] até TABELA [N] e trocar com
|   |   TABELA [I]";
|   |   I ← I + 1;
|   fim enquanto;
fim refinamento;
```

```
refinamento de "procurar o menor elemento e trocar";
|   J ← I + 1;
|   repita
|   |   se TABELA [J] < TABELA [I]
|   |   |   então
|   |   |       "trocar TABELA [J] com TABELA [I]";
|   |   fim se;
|   |   J ← J + 1;
|   até J > N;
fim refinamento;
```

```
refinamento de "trocar TABELA [J] com TABELA [I]";
|   AUX ← TABELA [I];
|   TABELA [I] ← TABELA [J];
|   TABELA [J] ← AUX;
fim refinamento;
```

```
refinamento de "imprimir tabela ordenada";
|   I ← 1;
|   enquanto I ≤ N faça
|   |   imprima (TABELA [I]);
|   |   I ← I + 1;
|   fim enquanto;
fim refinamento;
```

A versão completa do algoritmo deve ser feita porque é conveniente para escrever a versão final em uma linguagem de programação (comandos para a máquina real).